



JTAG interface for GNU Debugger

M-CORE



# User Manual

Manual Version 1.01 for BDI2000



©1997-2002 by Abatron AG

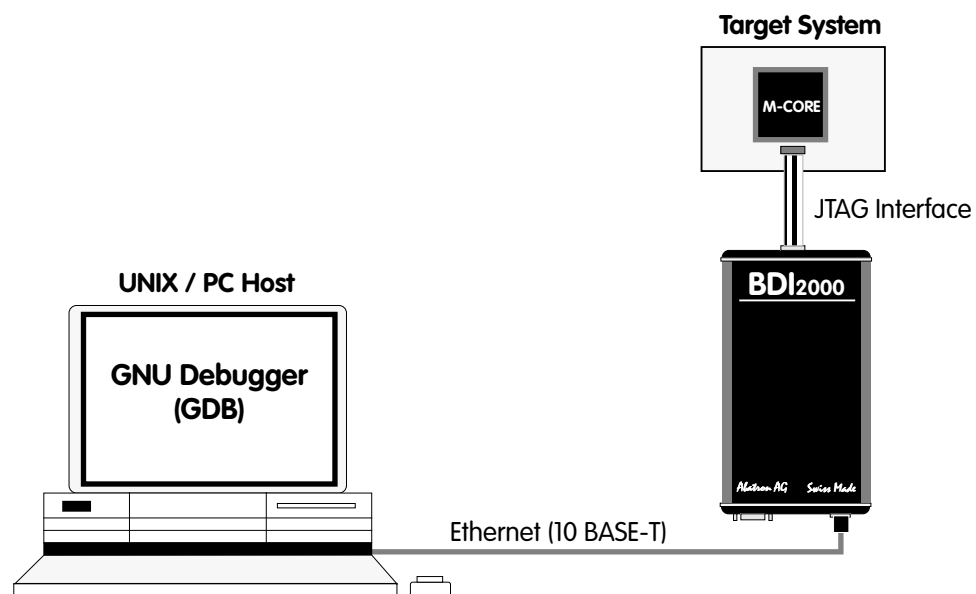
<b>1 Introduction .....</b>	<b>3</b>
1.1 BDI2000.....	3
1.2 BDI Configuration .....	4
<b>2 Installation .....</b>	<b>5</b>
2.1 Connecting the BDI2000 to Target.....	5
2.1.1 Changing Target Processor Type .....	7
2.2 Connecting the BDI2000 to Power Supply.....	8
2.2.1 External Power Supply .....	8
2.2.2 Power Supply from Target System .....	9
2.3 Status LED «MODE» .....	10
2.4 Connecting the BDI2000 to Host .....	11
2.4.1 Serial line communication .....	11
2.4.2 Ethernet communication .....	12
2.5 Installation of the Configuration Software .....	13
2.5.1 Configuration with a Linux / Unix host.....	14
2.5.2 Configuration with a Windows host .....	16
2.5.3 Recover procedure.....	17
2.6 Testing the BDI2000 to host connection .....	18
2.7 TFTP server for Windows NT.....	18
<b>3 Using bdiGDB .....</b>	<b>19</b>
3.1 Principle of operation .....	19
3.2 Configuration File .....	20
3.2.1 Part [INIT].....	21
3.2.2 Part [TARGET] .....	22
3.2.3 Part [HOST].....	23
3.2.4 Part [FLASH] .....	24
3.2.5 Part [REGS] .....	27
3.3 Debugging with GDB .....	29
3.3.1 Target setup.....	29
3.3.2 Connecting to the target.....	29
3.3.3 Breakpoint Handling.....	30
3.3.4 GDB monitor command.....	30
3.4 Telnet Interface .....	31
3.4.1 Command list .....	32
<b>4 Specifications .....</b>	<b>33</b>
<b>5 Environmental notice .....</b>	<b>34</b>
<b>6 Declaration of Conformity (CE).....</b>	<b>34</b>
<b>7 Warranty .....</b>	<b>35</b>
<b>7 Appendices</b>	
<b>A Troubleshooting .....</b>	<b>36</b>
<b>B Maintenance .....</b>	<b>37</b>
<b>C Trademarks .....</b>	<b>39</b>

## 1 Introduction

bdiGDB enhances the GNU debugger (GDB), with JTAG debugging for M-CORE based targets. With the builtin Ethernet interface you get a very fast download speed of up to 80 kBytes/sec. No target communication channel (e.g. serial line) is wasted for debugging purposes. Even better, you can use fast Ethernet debugging with target systems without network capability. The host to BDI communication uses the standard GDB remote protocol.

An additional Telnet interface is available for special debug tasks (e.g. force a hardware reset, program flash memory, ... ).

The following figure shows how the BDI2000 interface is connected between the host and the target:



### 1.1 BDI2000

The BDI2000 is the main part of the bdiGDB system. This small box implements the interface between the JTAG pins of the target CPU and a 10Base-T ethernet connector. The firmware and the programmable logic of the BDI2000 can be updated by the user with a simple Windows / Linux based configuration program. The BDI2000 supports 1.8 – 5.0 Volts target systems (3.0 – 5.0 Volts target systems with Rev. A/B).

## 1.2 BDI Configuration

As an initial setup, the IP address of the BDI2000, the IP address of the host with the configuration file and the name of the configuration file is stored within the flash of the BDI2000.

Every time the BDI2000 is powered on, it reads the configuration file via TFTP.

Following an example of a typical configuration file:

```
; bdiGDB configuration file MMCEVB2107 board
; -----
;
[INIT]
WM16 0x00C70000 0x000E ;WCR: disable watchdog
WM16 0x00C30000 0x2000 ;SYNCR: speed-up clock to 32MHz
WM16 0x00C20000 0x3303 ;CSCR0: external flash
WM16 0x00C20002 0x3303 ;CSCR1:
WM16 0x00C20004 0x3303 ;CSCR2: external SRAM
WM16 0x00C20006 0x3703 ;CSCR3:
WGPR 0 0x00000000 ;R0: Stackpointer = 0
;
WM32 0x00D00000 0x00000000 ;CMFRMCR: enable flash array
;WM32 0x00D00000 0x08000000 ;CMFRMCR: enable flash array, enable shadow information
;WM8 0x817FFFFD 0x10 ;MMIO: enable programming voltage
;DELAY 100 ;Delay after enable programming voltage

[TARGET]
CPUTYPE MMC2107
JTAGCLOCK 1 ;use 8 MHz JTAG clock
BDIMODE AGENT ;the BDI working mode (LOADONLY | AGENT)
BREAKMODE SOFT ;SOFT or HARD

[HOST]
IP 151.120.25.119
FILE E:\cygwin\home\bdidemo\mcore\testgnu.elf
FORMAT ELF
LOAD MANUAL ;load code MANUAL or AUTO after reset

[FLASH]
CHIPTYPE CMFR 32 ;program internal flash, cpu clock is 32MHz
;CHIPTYPE CMFRSHD 32 ;program shadow information, cpu clock is 32MHz
WORKSPACE 0x00800000 ;workspace in target RAM for fast programming algorithm
CHIPSIZE 0x20000 ;The size of one flash chip in bytes (e.g. AM29F040 = 0x80000)
BUSWIDTH 16 ;The width of the flash memory bus in bits (8 | 16 | 32)
FILE E:\cygwin\home\bdidemo\mcore\cmfr128.bin
FORMAT BIN 0x00000000
ERASE 0x000000FF ;Erase all CMFR blocks

[REGS]
FILE E:\cygwin\home\bdidemo\mcore\reg2107.def
```

Based on the information in the configuration file, the target is automatically initialized after every reset.

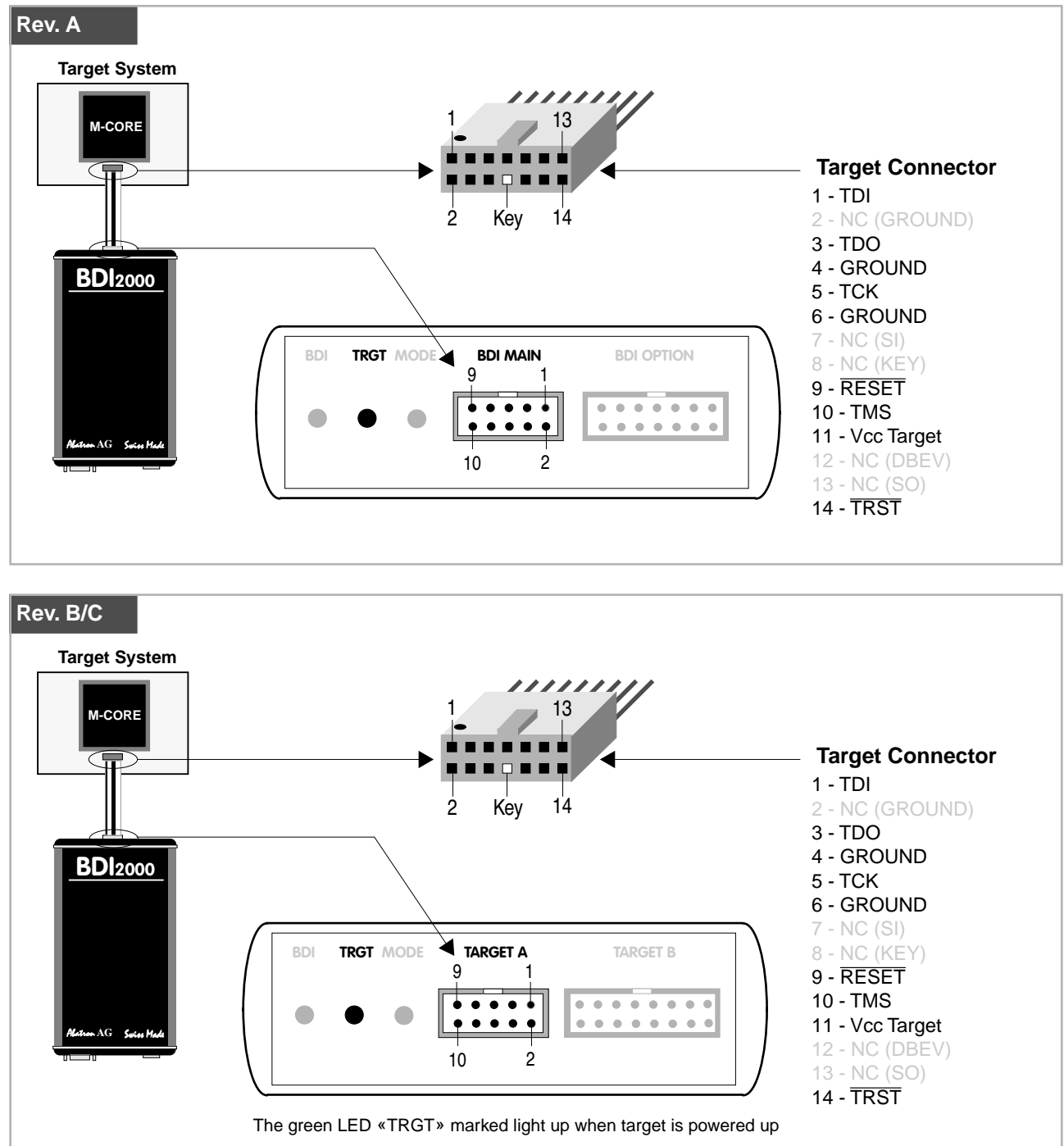
## 2 Installation

### 2.1 Connecting the BDI2000 to Target

The enclosed cable to the target system is designed for the standard JTAG/OnCE connector. In case where the target system has the same connector layout, the cable can be directly connected.



In order to ensure reliable operation of the BDI (EMC, runtimes, etc.) the target cable length must not exceed 20 cm (8").



For BDI MAIN / TARGET A connector signals see table on next page.

## BDI MAIN / TARGET A Connector Signals

Pin	Name	Description
1	(TMS1)	This pin is currently not used.
2	TRST	<b>JTAG Test Reset</b> This open collector output of the BDI2000 resets the JTAG TAP controller on the target.
3+5	GND	<b>System Ground</b>
4	TCK	<b>JTAG Test Clock</b> This output of the BDI2000 connects to the target TCK line.
6	TMS0	<b>JTAG Test Mode Select</b> This output of the BDI2000 connects to the target TMS line.
7	RESET	<b>Target Reset</b> This open collector output of the BDI2000 is used to reset the target system.
8	TDI	<b>JTAG Test Data In</b> This output of the BDI2000 connects to the target TDI line.
9	Vcc Target	<b>1.8 – 5.0V:</b> This is the target reference voltage. It indicates that the target has power and it is also used to create the logic-level reference for the input comparators. It also controls the output logic levels to the target. It is normally fed from Vdd I/O on the target board.  <b>3.0 – 5.0V with Rev. A/B :</b> This input to the BDI2000 is used to detect if the target is powered up. If there is a current limiting resistor between this pin and the target Vdd, it should be 100 Ohm or less.
10	TDO	<b>JTAG Test Data Out</b> This input to the BDI2000 connects to the target TDO line.

All the pins need to be connected to the target system for the debug operation.

### 2.1.1 Changing Target Processor Type

Before you can use the BDI2000 with an other target processor type (e.g. M-CORE <--> PPC), a new setup has to be done (see chapter 2.5). During this process the target cable must be disconnected from the target system. The BDI2000 needs to be supplied with 5 Volts via the BDI OPTION connector (Rev. A) or via the POWER connector (Rev. B/C). For more information see chapter 2.2.1 «External Power Supply»).



**To avoid data line conflicts, the BDI2000 must be disconnected from the target system while programming the logic for an other target CPU.**

## 2.2 Connecting the BDI2000 to Power Supply

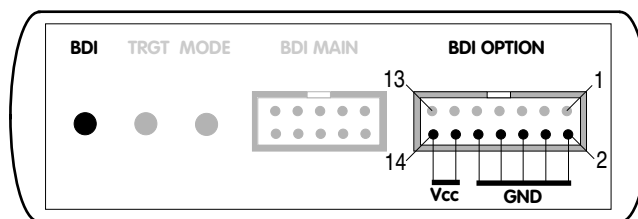
### 2.2.1 External Power Supply

The BDI2000 needs to be supplied with 5 Volts (max. 1A) via the BDI OPTION connector (Rev. A) or via POWER connector (Rev. B/C). The available power supply from Abatron (option) or the enclosed power cable can be directly connected. In order to ensure reliable operation of the BDI2000, keep the power supply cable as short as possible.



For error-free operation, the power supply to the BDI2000 must be between 4.75V and 5.25V DC. **The maximal tolerable supply voltage is 5.25 VDC. Any higher voltage or a wrong polarity might destroy the electronics.**

#### Rev. A

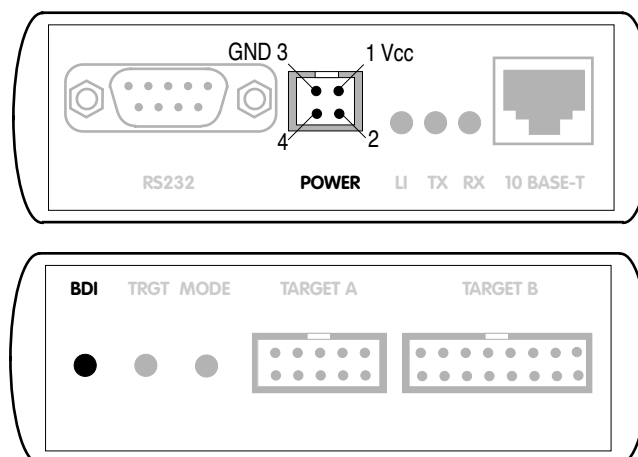


The green LED «BDI» marked light up when 5V power is connected to the BDI2000

#### BDI OPTION Connector

- 1 - NOT USED
- 2 - GROUND
- 3 - NOT USED
- 4 - GROUND
- 5 - NOT USED
- 6 - GROUND
- 7 - NOT USED
- 8 - GROUND
- 9 - NOT USED
- 10 - GROUND
- 11 - NOT USED
- 12 - Vcc (+5V)
- 13 - Vcc Target (+5V)
- 14 - Vcc (+5V)

#### Rev. B/C



The green LED «BDI» marked light up when 5V power is connected to the BDI2000

#### POWER Connector

- 1 - Vcc (+5V)
- 2 - VccTGT
- 3 - GROUND
- 4 - NOT USED

**Please switch on the system in the following sequence:**

- 1 --> external power supply
- 2 --> target system



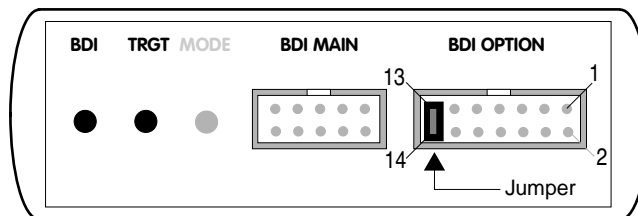
## 2.2.2 Power Supply from Target System

The BDI2000 needs to be supplied with 5 Volts (max. 1A) via BDI MAIN target connector (Rev. A) or via TARGET A connector (Rev. B/C). This mode can only be used when the target system runs with 5V and the pin «Vcc Target» is able to deliver a current up to 1A@5V. For pin description and layout see chapter 2.1 «Connecting the BDI2000 to Target». Insert the enclosed Jumper as shown in figure below. **Please ensure that the jumper is inserted correctly.**



For error-free operation, the power supply to the BDI2000 must be between 4.75V and 5.25V DC. **The maximal tolerable supply voltage is 5.25 VDC. Any higher voltage or a wrong polarity might destroy the electronics.**

### Rev. A

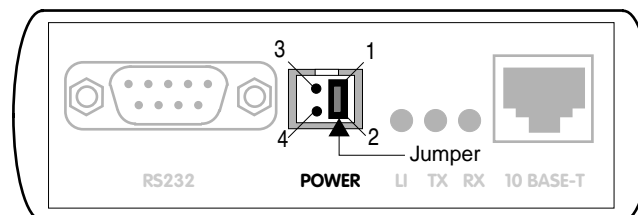


The green LEDs «BDI» and «TRGT» marked light up when target is powered up and the jumper is inserted correctly

### BDI OPTION Connector

- 1 - NOT USED
- 2 - GROUND
- 3 - NOT USED
- 4 - GROUND
- 5 - NOT USED
- 6 - GROUND
- 7 - NOT USED
- 8 - GROUND
- 9 - NOT USED
- 10 - GROUND
- 11 - NOT USED
- 12 - Vcc (+5V)
- 13 - Vcc Target (+5V)
- 14 - Vcc BDI2000 (+5V)

### Rev. B/C



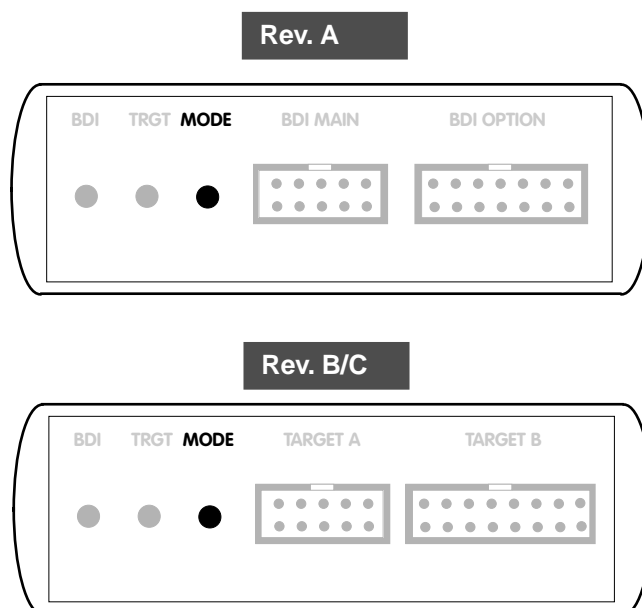
The green LEDs «BDI» and «TRGT» marked light up when target is powered up and the jumper is inserted correctly

### POWER Connector

- 1 - Vcc BDI2000 (+5V)
- 2 - Vcc Target (+5V)
- 3 - GROUND
- 4 - NOT USED

## 2.3 Status LED «MODE»

The built in LED indicates the following BDI states:



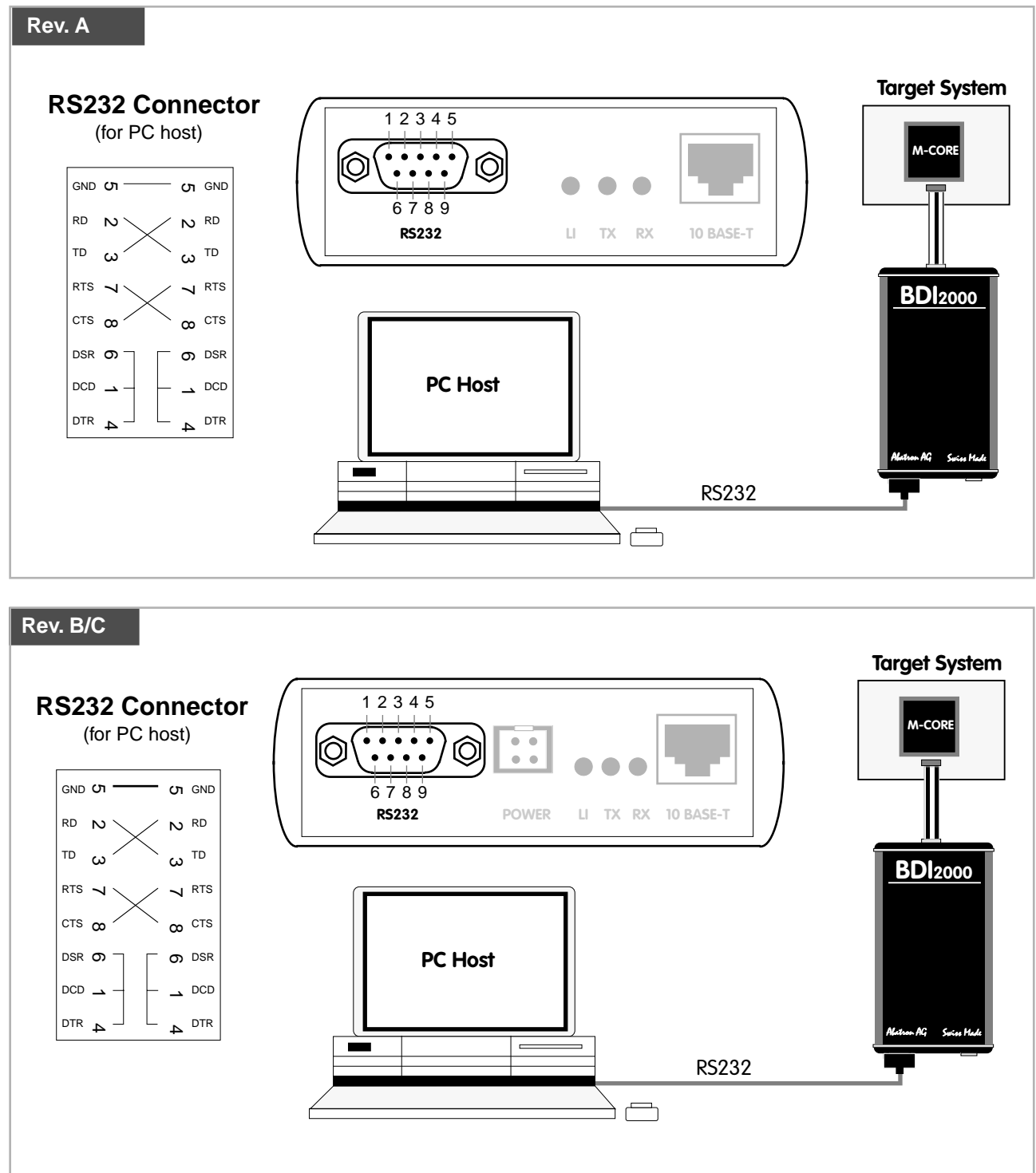
MODE LED	BDI STATES
OFF	The BDI is ready for use, the firmware is already loaded.
ON	The power supply for the BDI2000 is < 4.75VDC.
BLINK	The BDI «loader mode» is active (an invalid firmware is loaded or loading firmware is active).

## 2.4 Connecting the BDI2000 to Host

### 2.4.1 Serial line communication

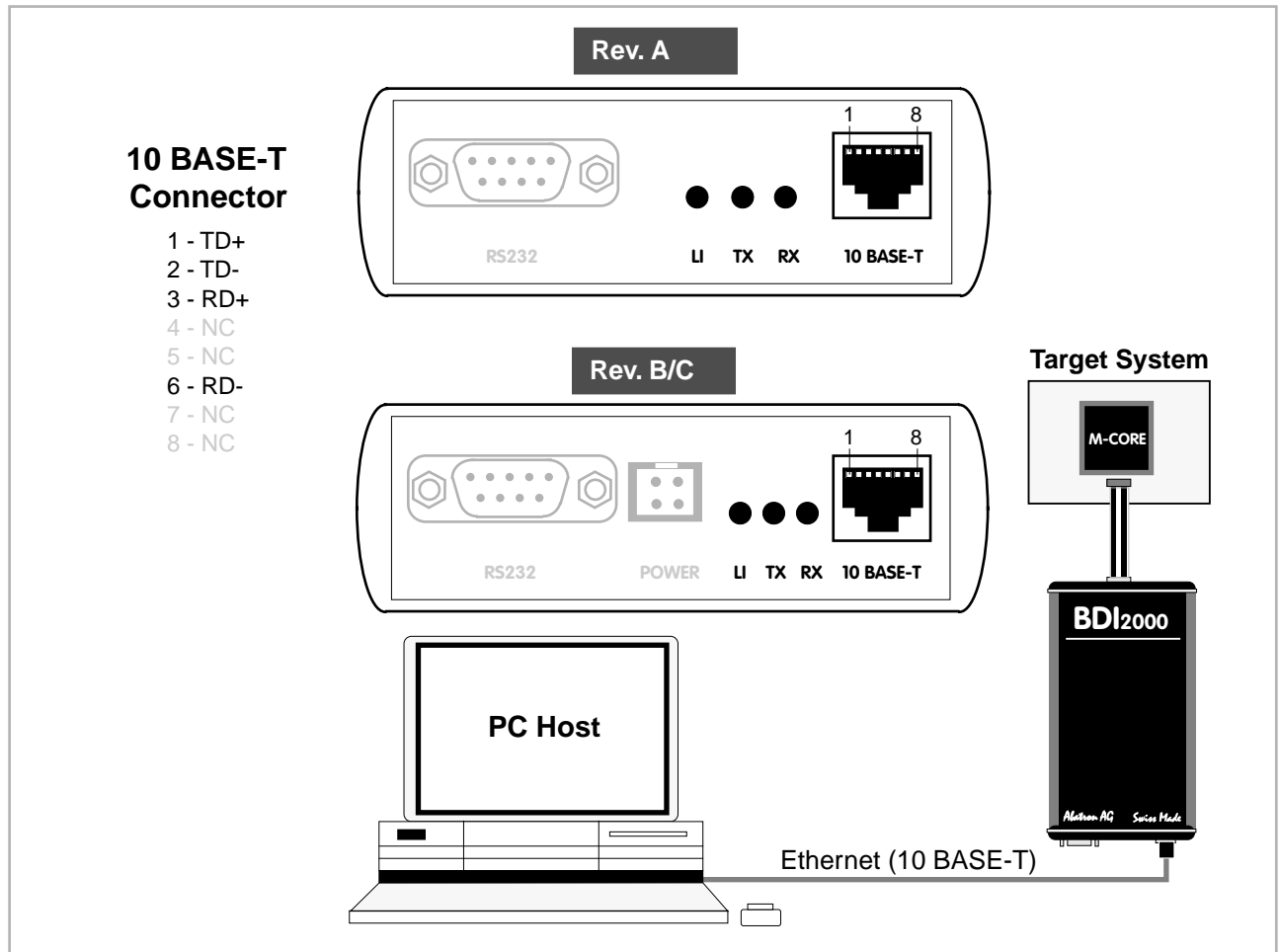
Serial line communication is only used for the initial configuration of the bdiGDB system.

The host is connected to the BDI through the serial interface (COM1...COM4). The communication cable (included) between BDI and Host is a serial cable. There is the same connector pinout for the BDI and for the Host side (Refer to Figure below).



## 2.4.2 Ethernet communication

The BDI2000 has a built-in 10 BASE-T Ethernet interface (see figure below). Connect an UTP (Unshielded Twisted Pair) cable to the BD2000. For thin Ethernet coaxial networks you can connect a commercially available media converter (BNC-->10 BASE-T) between your network and the BDI2000. Contact your network administrator if you have questions about the network.



The following explains the meanings of the built-in LED lights:

LED	Name	Description
LI	Link	When this LED light is ON, data link is successful between the UTP port of the BDI2000 and the hub to which it is connected.
TX	Transmit	When this LED light BLINKS, data is being transmitted through the UTP port of the BDI2000
RX	Receive	When this LED light BLINKS, data is being received through the UTP port of the BDI2000

## 2.5 Installation of the Configuration Software

On the enclosed diskette you will find the BDI configuration software and the firmware required for the BDI2000. For Windows NT users there is also a TFTP server included.

The following files are on the diskette.

b20mmcgd.exe	Windows configuration program
b20mmcgd.hlp	Windows help file for the configuration program
b20mmcgd.xxx	Firmware for the BDI2000
mmcjed20.xxx	JEDEC file for the BDI2000 (Rev. A/B) logic device when working with a M-CORE target
mmcjed21.xxx	JEDEC file for the BDI2000 (Rev. C) logic device when working with a M-CORE target
tftpsrv.exe	TFTP server for WindowsNT/ Windows95 (WIN32 console application)
*.cfg	Configuration files
*.def	Register definition files
bdisetup.zip	ZIP Archive with the Setup Tool sources for Linux / UNIX hosts.

### Overview of an installation / configuration process:

- Create a new directory on your hard disk
- Copy the entire contents of the enclosed diskette into this directory
- Linux only: extract the setup tool sources and build the setup tool
- Use the setup tool to load/update the BDI firmware/logic  
**Note:** A new BDI has no firmware/logic loaded.
- Use the setup tool to transmit the initial configuration parameters
  - IP address of the BDI.
  - IP address of the host with the configuration file.
  - Name of the configuration file. This file is accessed via TFTP.
  - Optional network parameters (subnet mask, default gateway).

## 2.5.1 Configuration with a Linux / Unix host

The firmware / logic update and the initial configuration of the BDI2000 is done with a command line utility. In the ZIP Archive bdisetup.zip are all sources to build this utility. More information about this utility can be found at the top in the bdisetup.c source file. There is also a make file included. Starting the tool without any parameter displays information about the syntax and parameters.



**To avoid data line conflicts, the BDI2000 must be disconnected from the target system while programming the logic for an other target CPU (see Chapter 2.1.1).**

Following the steps to bring-up a new BDI2000:

### 1. Build the setup tool:

The setup tool is delivered only as source files. This allows to build the tool on any Linux / Unix host. To build the tool, simply start the make utility.

```
[root@LINUX_1 bdisetup]# make
cc -O2 -c -o bdisetup.o bdisetup.c
cc -O2 -c -o bdicnf.o bdicnf.c
cc -O2 -c -o bdidll.o bdidll.c
cc -s bdisetup.o bdicnf.o bdidll.o -o bdisetup
```

### 2. Check the serial connection to the BDI:

With "bdisetup -v" you may check the serial connection to the BDI. The BDI will respond with information about the current loaded firmware and network configuration.

**Note:** Login as root, otherwise you probably have no access to the serial port.

```
[root@LINUX_1 bdisetup]# ./bdisetup -v -p/dev/ttyS0 -b57
BDI Type : BDI2000 Rev.C (SN: 92152150)
Loader   : V1.05
Firmware : unknown
Logic    : unknown
MAC      : ff-ff-ff-ff-ff-ff
IP Addr  : 255.255.255.255
Subnet   : 255.255.255.255
Gateway  : 255.255.255.255
Host IP  : 255.255.255.255
Config   : ??????????????????
```

### 3. Load/Update the BDI firmware/logic:

With "bdisetup -u" the firmware is loaded and the CPLD within the BDI2000 is programmed. This configures the BDI for the target you are using. Based on the parameters -a and -t, the tool selects the correct firmware / logic files. If the firmware / logic files are in the same directory as the setup tool, there is no need to enter a -d parameter.

```
[root@LINUX_1 bdisetup]# ./bdisetup -u -p/dev/ttyS0 -b57 -aGDB -tMCORE
Connecting to BDI loader
Erasing CPLD
Programming firmware with ./b20mmcgd.100
Programming CPLD with ./mmcjed21.100
```

#### 4. Transmit the initial configuration parameters:

With "bdisetup -c" the configuration parameters are written to the flash memory within the BDI. The following parameters are used to configure the BDI:

BDI IP Address	The IP address for the BDI2000. Ask your network administrator for assigning an IP address to this BDI2000. Every BDI2000 in your network needs a different IP address.
Subnet Mask	The subnet mask of the network where the BDI is connected to. A subnet mask of 255.255.255.255 disables the gateway feature. Ask your network administrator for the correct subnet mask. If the BDI and the host are in the same subnet, it is not necessary to enter a subnet mask.
Default Gateway	Enter the IP address of the default gateway. Ask your network administrator for the correct gateway IP address. If the gateway feature is disabled, you may enter 255.255.255.255 or any other value.
Config - Host IP Address	Enter the IP address of the host with the configuration file. The configuration file is automatically read by the BDI2000 after every start-up.
Configuration file	Enter the full path and name of the configuration file. This file is read via TFTP. Keep in mind that TFTP has it's own root directory (usual /tftpboot). You can simply copy the configuration file to this directory and the use the file name without any path. For more information about TFTP use "man tftpd".

```
[root@LINUX_1 bdisetup]# ./bdisetup -c -p/dev/ttyS0 -b57 \  
> -i151.120.25.101 \  
> -h151.120.25.118 \  
> -fmmc2107.cfg  
Connecting to BDI loader  
Writing network configuration  
Writing init list and mode  
Configuration passed
```

#### 5. Check configuration and exit loader mode:

The BDI is in loader mode when there is no valid firmware loaded or you connect to it with the setup tool. While in loader mode, the Mode LED is flashing. The BDI will not respond to network requests while in loader mode. To exit loader mode, the "bdisetup -v -s" can be used. You may also power-off the BDI, wait some time (1min.) and power-on it again to exit loader mode.

```
[root@LINUX_1 bdisetup]# ./bdisetup -v -p/dev/ttyS0 -b57 -s  
BDI Type : BDI2000 Rev.C (SN: 92152150)  
Loader : V1.05  
Firmware : V1.00 bdiGDB for M-CORE  
Logic : V1.00 M-CORE  
MAC : 00-40-49-fa-15-21  
IP Addr : 151.120.25.101  
Subnet : 255.255.255.255  
Gateway : 255.255.255.255  
Host IP : 151.120.25.118  
Config : mmc2107.cfg
```

The Mode LED should go off, and you can try to connect to the BDI via Telnet.

```
[root@LINUX_1 bdisetup]# telnet 151.120.25.101
```

## 2.5.2 Configuration with a Windows host

First make sure that the BDI is properly connected (see Chapter 2.1 to 2.4).



**To avoid data line conflicts, the BDI2000 must be disconnected from the target system while programming the logic for an other target CPU (see Chapter 2.1.1).**

*dialog box «BDI2000 Update/Setup»*

Before you can use the BDI2000 together with the GNU debugger, you must store the initial configuration parameters in the BDI2000 flash memory. The following options allow you to do this:

- |          |  |
|----------|--|
| Channel  | Select the communication port where the BDI2000 is connected during this setup session.  |
| Baudrate | Select the baudrate used to communicate with the BDI2000 loader during this setup session.   |
| Connect  | Click on this button to establish a connection with the BDI2000 loader. Once connected, the BDI2000 remains in loader mode until it is restarted or this dialog box is closed.   |
| Current  | Press this button to read back the current loaded BDI2000 software and logic versions. The current loader, firmware and logic version will be displayed.   |
| Update   | This button is only active if there is a newer firmware or logic version present in the execution directory of the bdiGDB setup software. Press this button to write the new firmware and/or logic into the BDI2000 flash memory / programmable logic. |

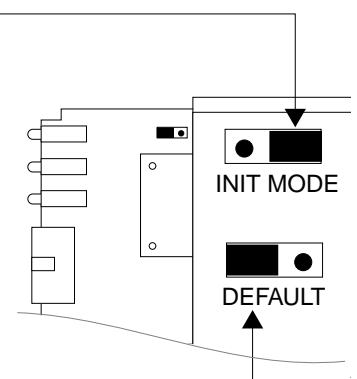


BDI IP Address	Enter the IP address for the BDI2000. Use the following format: xxx.xxx.xxx.xxx e.g.151.120.25.101 Ask your network administrator for assigning an IP address to this BDI2000. Every BDI2000 in your network needs a different IP address.
Subnet Mask	Enter the subnet mask of the network where the BDI is connected to. Use the following format: xxx.xxx.xxx.xx.e.g.255.255.255.0 A subnet mask of 255.255.255.255 disables the gateway feature. Ask your network administrator for the correct subnet mask.
Default Gateway	Enter the IP address of the default gateway. Ask your network administrator for the correct gateway IP address. If the gateway feature is disabled, you may enter 255.255.255.255 or any other value..
Config - Host IP Address	Enter the IP address of the host with the configuration file. The configuration file is automatically read by the BDI2000 after every start-up.
Configuration file	Enter the full path and name of the configuration file. e.g. D:\ada\target\config\bdi\evs332.cnf For information about the syntax of the configuration file see the bdiGDB User manual. This name is transmitted to the TFTP server when reading the configuration file.
Transmit	Click on this button to store the configuration in the BDI2000 flash memory.

### 2.5.3 Recover procedure

In rare instances you may not be able to load the firmware in spite of a correctly connected BDI (error of the previous firmware in the flash memory). **Before carrying out the following procedure, check the possibilities in Appendix «Troubleshooting».** In case you do not have any success with the tips there, do the following:

- Switch OFF the power supply for the BDI and open the unit as described in Appendix «Maintenance»
- Place the jumper in the «**INIT MODE**» position
- Connect the power cable or target cable if the BDI is powered from target system
- Switch ON the power supply for the BDI again and wait until the LED «MODE» blinks fast
- Turn the power supply OFF again
- Return the jumper to the «**DEFAULT**» position
- Reassemble the unit as described in Appendix «Maintenance»



## 2.6 Testing the BDI2000 to host connection

After the initial setup is done, you can test the communication between the host and the BDI2000. There is no need for a target configuration file and no TFTP server is needed on the host.

- If not already done, connect the bdiGDB system to the network.
- Power-up the BDI2000.
- Start a Telnet client on the host and connect to the BDI2000 (the IP address you entered during initial configuration).
- If everything is okay, a sign on message like «BDI Debugger for XScale» should be displayed in the Telnet window.

## 2.7 TFTP server for Windows NT

The bdiGDB system uses TFTP to access the configuration file and to load the application program. Because there is no TFTP server bundled with Windows NT, Abatron provides a TFTP server application **tftpsrv.exe**. This WIN32 console application runs as normal user application (not as a system service).

Command line syntax: `tftpsrv [p] [w] [dRootDirectory]`

Without any parameter, the server starts in read-only mode. This means, only read access request from the client are granted. This is the normal working mode. The bdiGDB system needs only read access to the configuration and program files.

The parameter [p] enables protocol output to the console window. Try it.

The parameter [w] enables write accesses to the host file system.

The parameter [d] allows to define a root directory.

<code>tftpsrv p</code>	Starts the TFTP server and enables protocol output
<code>tftpsrv p w</code>	Starts the TFTP server, enables protocol output and write accesses are allowed.
<code>tftpsrv dC:\tftp\</code>	Starts the TFTP server and allows only access to files in C:\tftp and its subdirectories. As file name, use relative names. For example "bdi\mpc750.cfg" accesses "C:\tftp\bdi\mpc750.cfg"

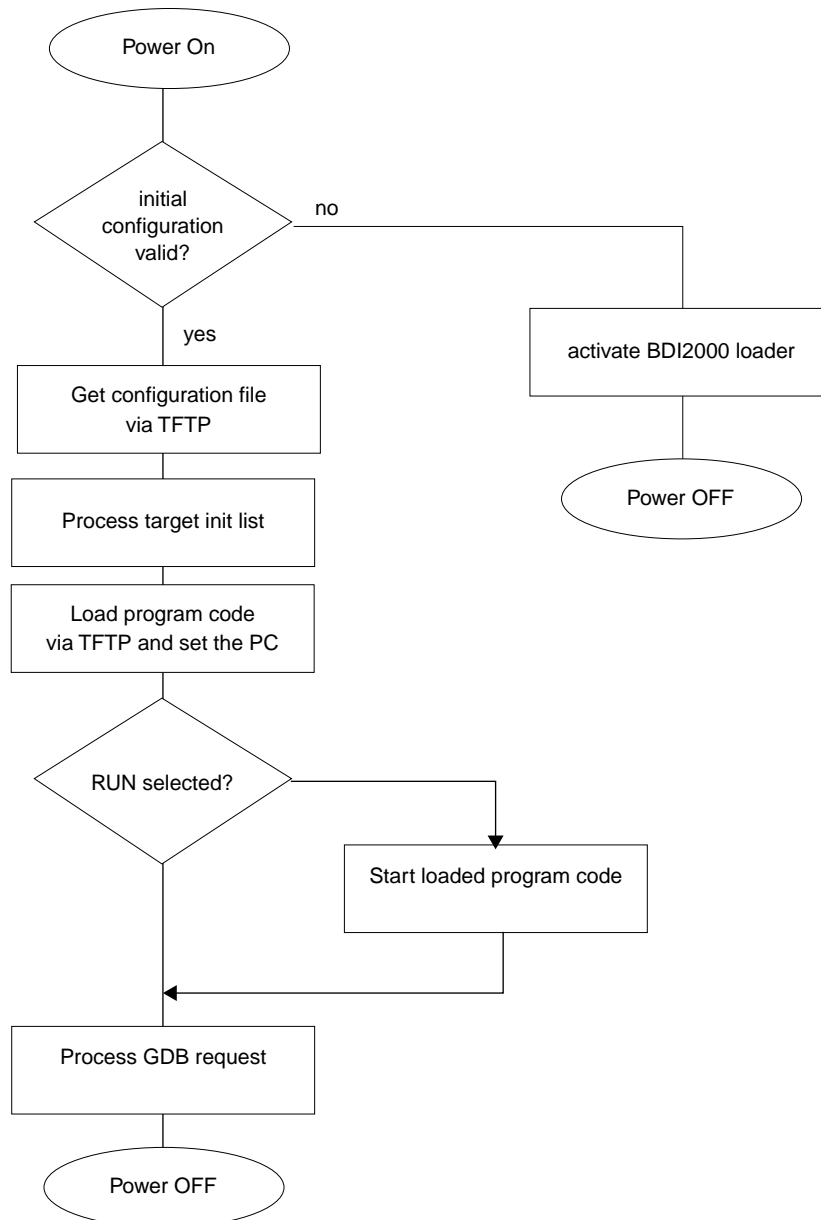
You may enter the TFTP server into the Startup group so the server is started every time you logon.

## 3 Using bdiGDB

### 3.1 Principle of operation

The firmware within the BDI handles the GDB request and accesses the target memory or registers via the JTAG interface. There is no need for any debug software on the target system. After loading the code via TFTP debugging can begin at the very first assembler statement.

Whenever the BDI system is powered-up the following sequence starts:



### 3.2 Configuration File

The configuration file is automatically read by the BDI2000 after every power on.  
The syntax of this file is as follows:

```
; comment
[part name]
identifier parameter1 parameter2 ..... parameterN ; comment
identifier parameter1 parameter2 ..... parameterN
.....
[part name]
identifier parameter1 parameter2 ..... parameterN
identifier parameter1 parameter2 ..... parameterN
.....
                etc.
```

Numeric parameters can be entered as decimal (e.g. 700) or as hexadecimal (0x80000).

### 3.2.1 Part [INIT]

The part [INIT] defines a list of commands which should be executed every time the target comes out of reset. The commands are used to get the target ready for loading the program file.

WGPR register value	<p>Write value to the selected general purpose register.</p> <p>register      the register number 0 .. 15</p> <p>value        the value to write into the register</p> <p>Example: WGPR 0 5</p>
WAFR register value	<p>Write value to the selected alternate register.</p> <p>register      the register number 0 .. 15</p> <p>value        the value to write into the register</p> <p>Example: WAFR 0 5</p>
WCR register value	<p>Write value to the selected control register.</p> <p>register      the register number 0 .. 12</p> <p>value        the value to write into the register</p> <p>Example: WCR 0 5</p>
WREG name value	<p>Write value to the selected CPU register by name</p> <p>name         the register name (PSR)</p> <p>value        the value to write into the register</p> <p>Example: WREG PSR 0x00000000</p>
WM8 address value	<p>Write a byte (8bit) to the selected memory place.</p> <p>address      the memory address</p> <p>value        the value to write to the target memory</p> <p>Example: WM8 0xFFFFFA21 0x04 ; SYPCR: watchdog disable ...</p>
WM16 address value	<p>Write a half word (16bit) to the selected memory place.</p> <p>address      the memory address</p> <p>value        the value to write to the target memory</p> <p>Example: WM16 0x00C70000 0x000E ; WCR: disable watchdog</p>
WM32 address value	<p>Write a word (32bit) to the selected memory place.</p> <p>address      the memory address</p> <p>value        the value to write to the target memory</p> <p>Example: WM32 0x02200000 0x01632440 ; SIUMCR</p>
DELAY value	<p>Delay for the selected time.</p> <p>value        the delay time in milliseconds (1...30000)</p> <p>Example: DELAY 500 ; delay for 0.5 seconds</p>

### 3.2.2 Part [TARGET]

The part [TARGET] defines some target specific values.

CPUTYPE type	This value gives the BDI information about the connected CPU.	
type	MMC2001, MMC2107, MMC2114	
Example:	CPUTYPE MMC2107	
JTAGCLOCK value	With this value you can select the JTAG clock rate the BDI2000 uses when communication with the target CPU.	
value	0 = 16.6 MHz	2 = 5.5 MHz
	1 = 8.3 MHz	3 = 4.1 MHz
Example:	JTAGCLOCK 1 ; JTAG clock is 8.3 MHz	
STARTUP mode [runtime]	This parameter selects the target startup mode. The following modes are supported:	
RESET	This default mode forces the target to debug mode immediately out of reset. No code is executed after reset.	
STOP	In this mode, the BDI lets the target execute code for "runtime" milliseconds after reset. This mode is useful when monitor code should initialize the target system.	
RUN	After reset, the target executes code until stopped by the Telnet "halt" command.	
Example:	STARTUP STOP 3000 ; let the CPU run for 3 seconds	
BDIMODE mode param	This parameter selects the BDI debugging mode. The following modes are supported:	
LOADONLY	Loads and starts the application code. No debugging via JTAG interface.	
AGENT	The debug agent runs within the BDI. There is no need for any debug software on the target. This mode accepts a second parameter. If RUN is entered as a second parameter, the loaded application will be started immediately, otherwise only the PC is set and BDI waits for GDB requests.	
Example:	BDIMODE AGENT RUN	
BREAKMODE mode param	This parameter defines how breakpoints are implemented.	
SOFT	This is the normal mode. Breakpoints are implemented by replacing code with a BKPT instruction. This mode accepts a second parameter. If NOADJ is entered as a second parameter, the BDI does not adjust the program counter back to the address of the BKPT instruction. This is necessary when using an old GDB version that adjusts the program counter it self (e.g. GDB 4.17 ).	
HARD	In this mode, the M-Core breakpoint hardware is used. Only 2 breakpoints at a time is supported.	
Example:	BREAKMODE HARD	

### 3.2.3 Part [HOST]

The part [HOST] defines some host specific values.

IP ipaddress	<p>The IP address of the host.</p> <p>ipaddress      the IP address in the form xxx.xxx.xxx.xxx</p> <p>Example:      IP 151.120.25.100</p>
FILE filename	<p>The file name of the program file. This name is used to access the application file via TFTP.</p> <p>filename      the filename including the full path</p> <p>Example:      FILE F:\gnu\demo\xscale\test.elf</p>
FORMAT format [offset]	<p>The format of the image file and an optional load address offset. If the image is already stored in ROM on the target, select ROM as the format. The optional parameter "offset" is added to any load address read from the image file.</p> <p>format          SREC, BIN, AOUT, ELF or ROM</p> <p>Example:      FORMAT ELF</p> <p>                 FORMAT ELF 0x10000</p>
LOAD mode	<p>In Agent mode, this parameters defines if the code is loaded automatically after every reset.</p> <p>mode            AUTO, MANUAL</p> <p>Example:      LOAD MANUAL</p>
START address	<p>The address where to start the program file. If this value is not defined and the core is not in ROM, the address is taken from the code file. If this value is not defined and the core is already in ROM, the PC will not be set before starting the target. This means, the program starts at the normal reset address (0x00000000).</p> <p>address        the address where to start the program file</p> <p>Example:      START 0x10000</p>
DEBUGPORT port	<p>The TCP port GDB uses to access the target.</p> <p>port            the TCP port number (default = 2001)</p> <p>Example:      DEBUGPORT 2001</p>
PROMPT string	<p>This entry defines a new Telnet prompt. The current prompt can also be changed via the Telnet interface.</p> <p>Example:      PROMPT mmc2107&gt;</p>
DUMP filename	<p>The default file name used for the Telnet DUMP command.</p> <p>filename        the filename including the full path</p> <p>Example:      DUMP dump.bin</p>

For a complete example see chapter «Introduction» or the file mmc2107.cfg on the distribution disk.

### 3.2.4 Part [FLASH]

The Telnet interface supports programming and erasing of flash memories. The bdiGDB system has to know which type of flash is used, how the chip(s) are connected to the CPU and which sectors to erase in case the ERASE command is entered without any parameter.

CHIPTYPE type [clock]	<p>This parameter defines the type of flash used. It is used to select the correct programming algorithm.</p> <p>type            AM29F, AM29BX8, AM29BX16, I28BX8, I28BX16, AT49, AT49X8, AT49X16, CMFR, CMFRSHD, SGFM</p> <p>clock           For the CMFR and SGFM flash, the BDI needs to know how fast the CPU is clocked. Enter the correct value in MHz. The allowed range is 8 ... 33 MHz for CMFR and 1...100 MHz for SGFM.</p> <p>Example:        CHIPTYPE    CMFR 32 ; internal flash, 32MHz</p>
CHIPSIZE size	<p>The size of <b>one</b> flash chip in bytes (e.g. AM29F010 = 0x20000). This value is used to calculate the starting address of the current flash memory bank.</p> <p>size            the size of one flash chip in bytes</p> <p>Example:        CHIPSIZE    0x80000</p>
BUSWIDTH width	<p>Enter the width of the memory bus that leads to the flash chips. Do not enter the width of the flash chip itself. The parameter CHIPTYPE carries the information about the number of data lines connected to one flash chip. For example, enter 16 if you are using two AM29F010 to build a 16bit flash memory bank.</p> <p>with            the width of the flash memory bus in bits (8   16   32)</p> <p>Example:        BUSWIDTH    16</p>
FILE filename	<p>The name of the file to program into the flash. This name is used to access the file via TFTP. This name may be overridden interactively at the Telnet interface.</p> <p>filename        the filename including the full path</p> <p>Example:        FILE    F:\gnu\mcore\bootrom.hex</p>
FORMAT format [offset]	<p>The format of the file and an optional address offset. The optional parameter "offset" is added to any load address read from the program file.</p> <p>format           SREC, BIN, AOUT or ELF</p> <p>Example:        FORMAT SREC</p> <p>                  FORMAT ELF 0x10000</p>



**WORKSPACE address** If a workspace is defined, the BDI uses a faster programming algorithm that runs out of RAM on the target system. Otherwise, the algorithm is processed within the BDI. The workspace is used for a 1kByte data buffer and to store the algorithm code. There must be at least 2kBytes of RAM available for this purpose. Programming internal flash also needs a workspace in target RAM

address the address of the RAM area

Example: WORKSPACE 0x00800000

**ERASE address [mode]** The flash memory may be individually erased via the Telnet interface. In order to make erasing of multiple flash sectors easier, you can enter an erase list. All entries in the erase list will be processed if you enter ERASE at the Telnet prompt without any parameter. For internal flash, this parameter has a different meaning. See below.

address Address of the flash sector, block or chip to erase

mode BLOCK, CHIP

Without this optional parameter, the BDI executes a sector erase. If supported by the chip, you can also specify a block or chip erase.

Example: ERASE 0x05040000 ;erase sector 4 of flash

ERASE 0x05060000 ;erase sector 6 of flash

ERASE 0x05000000 CHIP ;erase whole chip(s)

For the MMC2107 internal flash, the BDI assumes the following structure of the address:

16 bit	8 bit	8 bit
module address	<reserved>	block [0:7]

**module address** The 16 most significant bits of the flash module address.

**block** The bit mask to select the flash block to erase. Bit ordering is the same as in the CMFRCTL register (see MMC2107 manual).

```
[FLASH]
CHIPTYPE      CMFR 32      ;program internal flash, cpu clock is 32MHz
;CHIPTYPE     CMFRSHD 32   ;program shadow information,  cpu clock is 32MHz
WORKSPACE     0x00800000   ;workspace in target RAM for fast programming algorithm
CHIPSIZE      0x20000      ;The size of one flash chip in bytes (e.g. AM29F040 = 0x80000)
BUSWIDTH      16          ;The width of the flash memory bus in bits (8 | 16 | 32)
ERASE         0x000000FF   ;Erase all CMFR blocks
```

For SGFM flash (MMC2114, MMC2113) you can erase a block or a page:

```
CHIPTYPE      SGFM 32      ;program internal flash, system clock is 32MHz
WORKSPACE     0x00800000   ;workspace in target RAM for fast programming algorithm
ERASE         0x00000000 BLOCK ;erase 128k block 0 in MMC2114
ERASE         0x00020000 BLOCK ;erase 128k block 1 in MMC2114
```

### Supported Flash Memories:

There are currently 3 standard flash algorithm supported. The AMD, Intel and Atmel AT49 algorithm. Almost all currently available flash memories can be programmed with one of this algorithm. The flash type selects the appropriate algorithm and gives additional information about the used flash.

For 8bit only flash, select: AM29F, I28BX8 or AT49

For 8/16 bit flash in 8bit mode, select: AM29BX8, I28BX8 or AT49X8

For 8/16 bit flash in 16bit mode, select: AM29BX16, I28BX16 or AT49X16

For 16bit only flash, select: AM29BX16, I28BX16 or AT49X16

The AMD and AT49 algorithm are almost the same. The only difference is, that the AT49 algorithm does not check for the AMD status bit 5 (Exceeded Timing Limits).

Only the AMD and AT49 algorithm support chip erase. Block erase is only supported with the AT49 algorithm. If the algorithm does not support the selected mode, sector erase is performed. If the chip does not support the selected mode, erasing will fail. The erase command sequence is different only in the 6th write cycle. Depending on the selected mode, the following data is written in this cycle (see also flash data sheets): 0x10 for chip erase, 0x30 for sector erase, 0x50 for block erase.

The following table shows some examples:

Flash	x 8	x 16	Chipsize
Am29F010	AM29F	-	0x020000
Am29F800B	AM29BX8	AM29BX16	0x100000
Am29DL323C	AM29BX8	AM29BX16	0x400000
Intel 28F032B3	I28BX8	-	0x400000
Intel 28F640J3A	I28BX8	I28BX16	0x800000
Intel 28F320C3	-	I28BX16	0x400000
AT49BV040	AT49	-	0x080000
AT49BV1614	AT49X8	AT49X16	0x200000
SST39VF160	-	AT49X16	0x200000
MMC2107 internal flash	-	CMFR	-

### Note:

Some Intel flash chips (e.g. 28F800C3, 28F160C3, 28F320C3) power-up with all blocks in locked state. In order to erase/program those flash chips, use the init list to unlock the appropriate blocks.

```

WM16  0xFFFF0000    0x0060    unlock block 0
WM16  0xFFFF0000    0x00D0
WM16  0xFFFF10000   0x0060    unlock block 1
WM16  0xFFFF10000   0x00D0
      . . . .
WM16  0xFFFF0000    0xFFFF    select read mode

```

### 3.2.5 Part [REGS]

In order to make it easier to access target registers via the Telnet interface, the BDI can read in a register definition file. In this file, the user defines a name for the register and how the BDI should access it (e.g. as memory mapped, memory mapped with offset, ...). The name of the register definition file and information for different registers type has to be defined in the configuration file.

The register name, type, address/offset/number and size are defined in a separate register definition file. This way, you can create one register definition file for a specific target processor that can be used for all possible positions of the internal memory map. You only have to change one entry in the configuration file.

An entry in the register definition file has the following syntax:

name    type    addr    size

name	The name of the register (max. 12 characters)	
type	The register type	
	GPR	General purpose register
	AFR	Alternate register
	CR	Control register
	MM	Absolute direct memory mapped register
	DMM1...DMM4	Relative direct memory mapped register
	IMM1...IMM4	Indirect memory mapped register
addr	The address, offset or number of the register	
size	The size (8, 16, 32) of the register, default is 32	

The following entries are supported in the [REGS] part of the configuration file:

FILE filename	The name of the register definition file. This name is used to access the file via TFTP. The file is loaded once during BDI startup.		
	filename	the filename including the full path	
	Example:	FILE C:\bdi\regs\reg2107.def	
DMMn base	This defines the base address of direct memory mapped registers. This base address is added to the individual offset of the register.		
	base	the base address	
	Example:	DMM1 0x01000	
IMMn addr data	This defines the addresses of the memory mapped address and data registers of indirect memory mapped registers. The address of a IMMn register is first written to "addr" and then the register value is access using "data" as address.		
	addr	the address of the Address register	
	data	the address of the Data register	
	Example:	DMM1 0x04700000	

## Example for a register definition:

Entry in the configuration file:

```
[REGS]
FILE      E:\cygwin\home\bdidemo\mcore\reg2107.def
```

The register definition file:

```
;
;name          type   addr          size
;-----
;
;      Ports
porta  MM       0x00c00000      8
portb  MM       0x00c00001      8
;
;      .....
clrh   MM       0x00c0002b      8
clri   MM       0x00c0002c      8
;
pcdpar MM       0x00c00030      8
pepar  MM       0x00c00031      8
;
;      Chip Configuration
ccr     MM       0x00c10000     16
rcon    MM       0x00c10004     16
cir     MM       0x00c10006     16
ctr     MM       0x00c10008     16
;
;      Chip Selects
cscr0   MM       0x00c20000     16
cscr1   MM       0x00c20002     16
cscr2   MM       0x00c20004     16
cscr3   MM       0x00c20006     16
;
;      Clocks
syncr   MM       0x00c30000     16
synsr   MM       0x00c30002      8
syntr   MM       0x00c30003      8
syntr2  MM       0x00c30004     32
;
;      Reset
rcr     MM       0x00c40000      8
rsr     MM       0x00c40001      8
rtr     MM       0x00c40002      8
;
;      Interrupt Controller
icr     MM       0x00c50000     16
isr     MM       0x00c50002     16
ifrh    MM       0x00c50004     32
ifrl    MM       0x00c50008     32
ipr     MM       0x00c5000c     32
nier    MM       0x00c50010     32
nipr    MM       0x00c50014     32
fier    MM       0x00c50018     32
fipr    MM       0x00c5001c     32
;
;      Flash
cmfrmcr MM       0x00d00000     32
;
```

### 3.3 Debugging with GDB

Because the target agent runs within BDI, no debug support has to be linked to your application. There is also no need for any BDI specific changes in the application sources. Your application must be fully linked because no dynamic loading is supported.

#### 3.3.1 Target setup

Target initialization may be done at two places. First with the BDI configuration file, second within the application. The setup in the configuration file must at least enable access to the target memory where the application will be loaded. Disable the watchdog and setting the CPU clock rate may also be done with the BDI configuration file. Application specific initializations like setting the timer rate are best located in the application startup sequence.

#### 3.3.2 Connecting to the target

As soon as the target comes out of reset, BDI initializes it and loads your application code. If RUN is selected, the application is immediately started, otherwise only the target PC is set. BDI now waits for GDB request from the debugger running on the host.

After starting the debugger, it must be connected to the remote target. This can be done with the following command at the GDB prompt:

```
(gdb)target remote bdi2000:2001
```

bdi2000                      This stands for an IP address. The HOST file must have an appropriate entry. You may also use an IP address in the form xxx.xxx.xxx.xxx

2001                        This is the TCP port used to communicate with the BDI

If not already suspended, this stops the execution of application code and the target CPU changes to background debug mode.

Remember, every time the application is suspended, the target CPU is freezed. During this time no hardware interrupts will be processed.

**Note:** For convenience, the GDB detach command triggers a target reset sequence in the BDI.

```
(gdb)...
```

```
(gdb)detach
```

```
... Wait until BDI has resetet the target and reloaded the image
```

```
(gdb)target remote bdi2000:2001
```

**Note:**

After loading a program to the target you cannot use the GDB "*run*" command to start execution. You have to use the GDB "*continue*" command.

### 3.3.3 Breakpoint Handling

There are two breakpoint modes supported. One of them (SOFT) is implemented by replacing application code with a BKPT instruction. The other (HARD) uses the built in breakpoint logic. If HARD is selected, only up to 2 breakpoints can be active at the same time.

The following example selects SOFT as the breakpoint mode:

```
BREAKMODE    SOFT          ;SOFT or HARD, HARD uses hardware breakpoints
```

#### **GDB versions before V5.0:**

GDB inserts breakpoints by replacing code via simple memory read / write commands. There is no command like "Set Breakpoint" defined in the GDB remote protocol. When breakpoint mode HARD is selected, the BDI checks the memory write commands for such hidden "Set Breakpoint" actions. If such a write is detected, the write is not performed and the BDI sets an appropriate hardware breakpoint. The BDI assumes that this is a "Set Breakpoint" action when memory write length is 2 bytes and the pattern to write is the BKPT instruction.

#### **GDB version V5.x:**

GDB version 5.x uses the Z-packet to set breakpoints (watchpoints). For software breakpoints, the BDI replaces code with BKPT instruction. When breakpoint mode HARD is selected, the BDI sets an appropriate hardware breakpoint.

### 3.3.4 GDB monitor command

The BDI supports the GDB V5.x "monitor" command. Telnet commands are executed and the Telnet output is returned to GDB.

```
(gdb) target remote bdi2000:2001
Remote debugging using bdi2000:2001
0x10b2 in start ()
(gdb) monitor md 0 1
00000000 : 0xe59ff018 - 442503144 ...
```

### 3.4 Telnet Interface

A Telnet server is integrated within the BDI. The Telnet channel is used by the BDI to output error messages and other information. Also some basic debug tasks may be done by using this interface. Enter help at the Telnet command prompt to get a list of the available commands.

Telnet Debug features:

- Display and modify memory locations
- Display and modify registers
- Single step a code sequence
- Set hardware breakpoints (for code and data accesses)
- Load a code file from any host
- Start / Stop program execution
- Programming and Erasing Flash memory

During debugging with GDB, the Telnet is mainly used to reboot the target (generate a hardware reset and reload the application code). It may be also useful during the first installation of the bdiGDB system or in case of special debug needs.

Because the Telnet server within the BDI generates no echo, enable local echo at your Telnet client.

### 3.4.1 Command list

```
"MD      [<address>] [<count>]  display target memory as word (32bit)",
"MDH     [<address>] [<count>]  display target memory as half word (16bit)",
"MDB     [<address>] [<count>]  display target memory as byte (8bit)",
"DUMP    <addr> <size> [<file>] dump target memory to a file",
"MM      <addr> <value> [<cnt>] modify word(s) (32bit) in target memory",
"MMH     <addr> <value> [<cnt>] modify half word(s) (16bit) in target memory",
"MMB     <addr> <value> [<cnt>] modify byte(s) (8bit) in target memory",
"MC      [<address>] [<count>]  calculates a checksum over a memory range",
"MV      verifies the last calculated checksum",
"RD      [{<nbr>#<name>}]      display general purpose or user defined register",
"RDUMP   [<file>]             dump all user defined register to a file",
"RDA     display alternate register file",
"RDC     display control registers",
"RM      {<nbr>#<name>} <value> modify general purpose or user defined register",
"RMA     <number> <value>      modify alternate register",
"RMC     <number> <value>      modify control register",
"BOOT    reset the BDI and reload the configuration",
"RESET   [HALT | RUN [time]]  reset the target system, change startup mode",
"GO      [<pc>]               set PC and start target system",
"TI      [<pc>]               single step an instruction",
"HALT    force target to enter debug mode",
"BI      <from> [<to>] [<count>] set instruction hardware breakpoint",
"BD      [R|W] <addr> [<count>] set data watchpoint (32bit access)",
"BDH     [R|W] <addr> [<count>] set data watchpoint (16bit access)",
"BDB     [R|W] <addr> [<count>] set data watchpoint ( 8bit access)",
"BC      [<id>]               clear breakpoint(s)",
"INFO    display information about the current state",
"LOAD    [<offset>] [<file> [<format>]] load program file to target memory",
"VERIFY  [<offset>] [<file> [<format>]] verify a program file to target memory",
"PROG    [<offset>] [<file> [<format>]] program flash memory",
"        <format> : SREC or BIN or AOUT or ELF",
"ERASE   [<address> [<mode>]]  erase a flash memory sector, chip or block",
"        <mode> : CHIP, BLOCK or SECTOR (default is sector)",
"HOST    <ip>                 change IP address of program file host",
"PROMPT  <string>              defines a new prompt string",
"CONFIG  display BDI configuration",
"HELP    display command list",
"QUIT    terminate the Telnet session"
```

#### Notes:

The DUMP command uses TFTP to write a binary image to a host file. Writing via TFTP on a Linux/Unix system is only possible if the file already exists and has public write access. Use "man tftpd" to get more information about the TFTP server on your host.



## 4 Specifications

Operating Voltage Limiting	5 VDC $\pm$ 0.25 V
Power Supply Current	typ. 500 mA max. 1000 mA
RS232 Interface: Baud Rates	9'600, 19'200, 38'400, 57'600, 115'200
Data Bits	8
Parity Bits	none
Stop Bits	1
Network Interface	10 BASE-T
Serial Transfer Rate between BDI and Target	up to 16 Mbit/s
Supported target voltage	1.8 – 5.0 V (3.0 – 5.0 V with Rev. A/B)
Operating Temperature	+ 5 °C ... +60 °C
Storage Temperature	-20 °C ... +65 °C
Relative Humidity (noncondensing)	<90 %rF
Size	190 x 110 x 35 mm
Weight (without cables)	420 g
Host Cable length (RS232)	2.5 m




Specifications subject to change without notice

## 5 Environmental notice



Disposal of the equipment must be carried out at a designated disposal site.

## 6 Declaration of Conformity (CE)

  
**DECLARATION OF CONFORMITY**  
This declaration is valid for following product:  
**Type of device: BDM/JTAG Interface**  
**Product name: BDI2000**  
The signing authorities state, that the above mentioned equipment meets  
the requirements for emission and immunity according to  
**EMC Directive 89/336/EEC**  
The evaluation procedure of conformity was assured according to the  
following standards:  
**EN 50081-2**  
**EN 50082-2**  
This declaration of conformity is based on the test report no.  
QNL-E853-05-8-a of QUINEL, Zug, accredited according to EN 45001.  
Manufacturer:  
**ABATRON AG**  
**Stöckenstrasse 4**  
**CH-6221 Rickenbach**  
Authority:  
  
Max Vock  
Marketing Director  
  
Ruedi Dummermuth  
Technical Director  
Rickenbach, May 30, 1998

## 7 Warranty

ABATRON Switzerland warrants the physical diskette, cable, BDI2000 and physical documentation to be free of defects in materials and workmanship for a period of 24 months following the date of purchase when used under normal conditions.

In the event of notification within the warranty period of defects in material or workmanship, ABATRON will replace defective diskette, cable, BDI2000 or documentation. The remedy for breach of this warranty shall be limited to replacement and shall not encompass any other damages, including but not limited loss of profit, special, incidental, consequential, or other similar claims.

ABATRON Switzerland specifically disclaims all other warranties- expressed or implied, including but not limited to implied warranties of merchantability and fitness for particular purposes - with respect to defects in the diskette, cable, BDI2000 and documentation, and the program license granted herein, including without limitation the operation of the program with respect to any particular application, use, or purposes. In no event shall ABATRON be liable for any loss of profit or any other commercial damage, including but not limited to special, incidental, consequential, or other damages.

Failure in handling which leads to defects are not covered under this warranty. The warranty is void under any self-made repair operation except exchanging the fuse.

## Appendices

### A Troubleshooting

#### Problem

The firmware can not be loaded.

#### Possible reasons

- The BDI is not correctly connected with the target system (see chapter 2).
- The power supply of the target system is switched off or not in operating range (4.75 VDC ... 5.25 VDC) --> MODE LED is OFF or RED
- The built in fuse is damaged --> MODE LED is OFF
- The BDI is not correctly connected with the Host (see chapter 2).
- A wrong communication port (Com 1...Com 4) is selected.

#### Problem

No working with the target system (loading firmware is ok).

#### Possible reasons

- Wrong pin assignment (BDM/JTAG connector) of the target system (see chapter 2).
- Target system initialization is not correctly --> enter an appropriate target initialization list.
- An incorrect IP address was entered (BDI2000 configuration)
- BDM/JTAG signals from the target system are not correctly (short-circuit, break, ...).
- The target system is damaged.

#### Problem

Network processes do not function (loading the firmware was successful)

#### Possible reasons

- The BDI2000 is not connected or not correctly connected to the network (LAN cable or media converter)
- An incorrect IP address was entered (BDI2000 configuration)

## B Maintenance

The BDI needs no special maintenance. Clean the housing with a mild detergent only. Solvents such as gasoline may damage it.

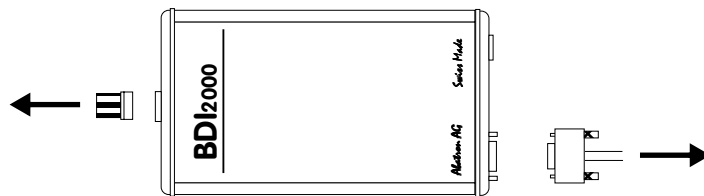
If the BDI is connected correctly and it is still not responding, then the built in fuse might be damaged (in cases where the device was used with wrong supply voltage or wrong polarity). To exchange the fuse or to perform special initialization, please proceed according to the following steps:



**Observe precautions for handling (Electrostatic sensitive device)**  
**Unplug the cables before opening the cover.**  
**Use exact fuse replacement (Microfuse MSF 1.6 AF).**

**1**

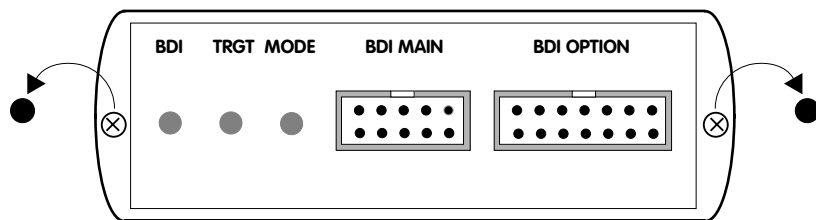
1.1 Unplug the cables



**2**

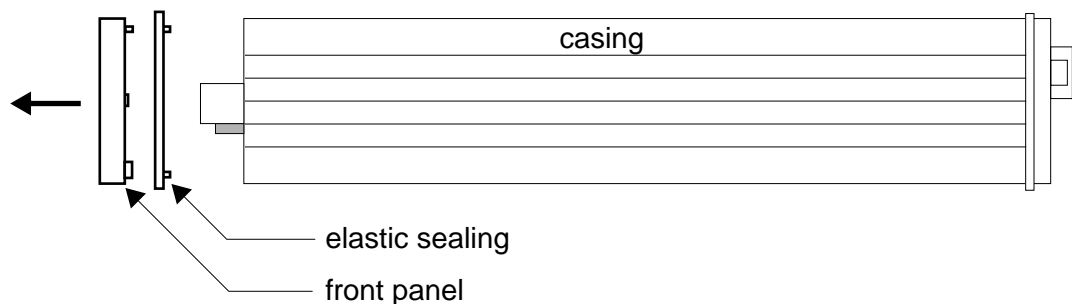
2.1 Remove the two plastic caps that cover the screws on target front side (e.g. with a small knife)

2.2 Remove the two screws that hold the front panel



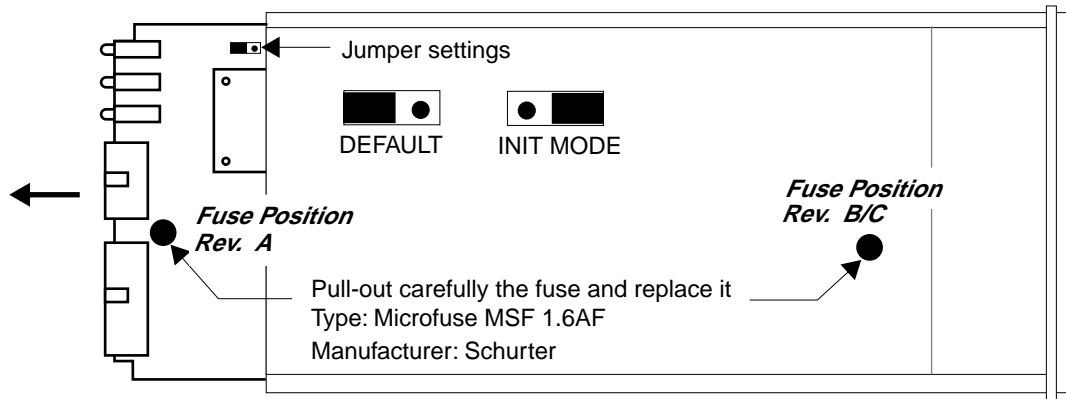
**3**

3.1 While holding the casing, remove the front panel and the red elastig sealing



4

4.1 While holding the casing, slide carefully the print in position as shown in figure below

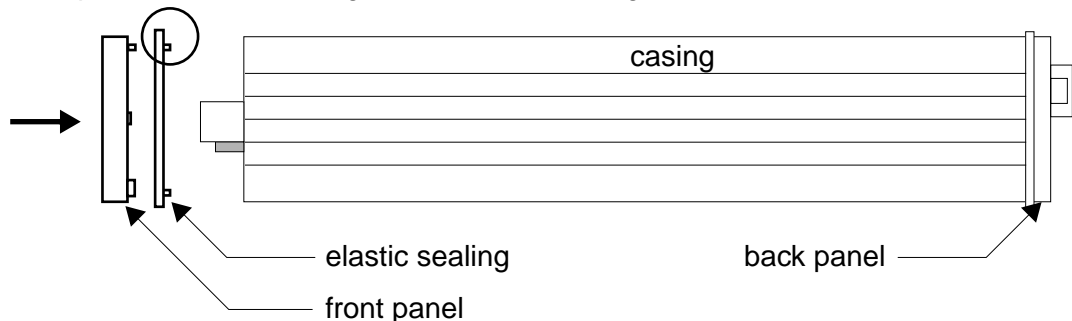


5

## Reinstallation

5.1 Slide back carefully the print. Check that the LEDs align with the holes in the back panel.

5.2 Push carefully the front panel and the red elastic sealing on the casing. Check that the LEDs align with the holes in the front panel and that the position of the sealing is as shown in the figure below.



5.3 Mount the screws (do not overtighten it)

5.4 Mount the two plastic caps that cover the screws

5.5 Plug the cables



**Observe precautions for handling (Electrostatic sensitive device)**  
**Unplug the cables before opening the cover.**  
**Use exact fuse replacement (Microfuse MSF 1.6 AF).**

## **C Trademarks**

All trademarks are property of their respective holders.